REMARKS

This Application has been carefully reviewed in light of the Office Action mailed July 2, 2004 ("Office Action"). At the time of the Office Action, Claims 1-37 were pending in the application. In the Office Action, the Examiner rejects Claims 1-37. Applicants amend Claims 1, 13, 23, 29, 35 and 37. Applicants respectfully request reconsideration and favorable action in this case. Applicants note with appreciation the Examiner's acceptance of the drawings filed on July 5, 2001.

Interview Summary

Applicants thank the Examiner for conducting the telephone interview on September 21, 2004, and for the thoughtful consideration of this case. During the telephone interview, Applicants and Examiner discussed the patentability of independent Claim 1 over IBM, "Software Compiler for Analysis and Measuring Programs," 9/1993 ("*Scamp*"), U.S. Patent No. 5,729,746 issued to Leonard ("*Leonard*"), and U.S. Patent No. 6,529,865 issued to Duan et al. ("*Duan*"). It is Applicants' understanding that no agreement was reached regarding independent Claim 1.

Section 112 Rejections

The Examiner rejects Claim 23 under 35 U.S.C. § 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which Applicants regard as the invention. Specifically, the Examiner states that "the parser" of Claim 23 lacks antecedent basis. Applicants have amended Claim 23 to correct the antecedent basis issue identified by the Examiner. Accordingly, Applicants respectfully submit that Claim 23 is in accordance with 35 U.S.C. § 112, second paragraph. Therefore, Applicants respectfully request reconsideration and allowance of Claim 23.

Section 103 Rejections

The Examiner rejects Claims 1-7, 13-18, 23-26, and 37 under 35 U.S.C. § 103(a) as being unpatentable over *Scamp* in view of *Leonard*, and further in view of *Duan*. The Examiner rejects Claims 8-11, 19-21, and 27-28 under 35 U.S.C. § 103(a) as being unpatentable over *Scamp*, in view of *Leonard*, and *Duan*, and further in view of U.S. Patent

No. 4,931,928 issued to Greenfeld ("*Greenfeld*"). The Examiner rejects Claims 12, 22, and 29-36 under 35 U.S.C. § 103(a) as being unpatentable over *Scamp*, in view of *Leonard*, *Duan*, and *Greenfeld*, and further in view of U.S. Patent No. 3,711,863 issued to Bloom ("*Bloom*"). For the following reasons, Applicants respectfully request reconsideration and allowance of Claims 1-37.

<u>The Claims are Allowable over the Cited References</u>

To defeat a patent under 35 U.S.C. § 103, "the prior art references must teach or suggest all the claim limitations." *In re Vaeck*, 947 F.2d 488 (Fed. Cir. 1991); M.P.E.P. § 706.02(j). Applicants respectfully submit that the proposed combinations of references do not disclose, teach, or suggest each and every element recited in Applicants' claims.

For example, the Examiner relies on the *Scamp-Leonard-Duan* combination in rejecting Applicants' independent Claim 1. Claim 1 recites a source code line counting system that includes a software counting tool, which is operable to:

- create a token stream,
- create a list of statements in response to the token stream, at least some of the statements comprising a combination of two or more tokens, and
- generate a count value in response to the list of statements.

In the Office Action, the Examiner relies on *Scamp* for disclosure of "creat[ing] a token stream" and *Leonard* for disclosure of "creat[ing] a list of statements in response to the token stream." Neither reference, however, discloses, teaches, or suggest the specific combination of features recited in Applicants' Claim 1.

With regard to the primary reference relied upon by the Examiner, *Scamp* merely discloses a software compiler for analyzing and measuring programs. (Title). The software compiler of *Scamp* defines tokens and places flags before tokens that are to be measured. (Page 127). The output of the software compiler is a report. (Page 24). *Scamp* provides an example of the type of metrics which may be output by the software compiler. "For example, *Scamp* measures how many times the START token is flagged as an OPERATOR independently of how many times the assignment operator is flagged as an OPERATOR." (Page 127). Therefore, *Scamp* merely analyzes how often and how many times a token appears in the source code and, thus, merely provides a count of tokens. Accordingly, Applicants submit (and the Examiner has not suggested otherwise) that *Scamp* does not disclose, teach, or suggest a software counting tool operable to "create a list of statements in response to the token stream" and "generate a count value in response to the list of statements," as recited in Claim 1. In fact, the claimed list of statements is completely absent from the teachings of *Scamp*. As such, *Scamp* certainly cannot be said to disclose, teach, or suggest "at least some of the statements comprising a combination of two or more tokens," as also recited in Claim 1.

The recited features are also absent from the disclosure of *Leonard*. *Leonard* merely discloses a tool for estimating the final size of a software product using a life-cycle model. (Title). To this end, "metrics are used as a quantitative measurement of the effort and cost of the problem solution within the framework of the model." (Column 2, lines 6-8). Thus, *Leonard* is a development tool for determining a predictor metric that may have a "strong correlation to some later result." (Column 3, lines 66-67). *Leonard* does not provide metrics relating to <u>developed</u> source code. Rather, *Leonard* generates tokens in anticipation of what the source code, when eventually written, will do.

Furthermore, similar to *Scamp*, *Leonard* also merely discloses counting tokens. Specifically, *Leonard* discloses that "[b]ecause some lines of code are more difficult to write than others, and since some languages allow multiple lines of code in a single line of text, difficulties arise in using the LOC [lines of code] metric in a uniform manner." (Column 4, lines 33-36). "One way to overcome these difficulties is to use a count of the number of tokens in the program as a metric of size." (Column 4, lines 36-38). The LOC metric may then be calculated as the number of tokens divided by some constant, which "depends on the language used." (Column 5, lines 17-22). Because the basis of the predictor metric, however is a count of tokens, *Leonard* also does not disclose, teach, or suggest a software counting tool operable to "create a list of statements in response to the token stream" and "generate a count value in response to the list of statements," as recited in Claim 1. Similar to *Scamp*, the claimed list of statements is completely absent from the disclosure of *Leonard*. As such, *Leonard* also cannot be said to disclose, teach, or suggest "at least some of the statements comprising a combination of two or more tokens," as also recited in Claim 1.

Applicants' invention produces more accurate measures of code complexity than prior art systems that count tokens. Counting tokens produces less accurate results due to variations in programmer style. During the interview, the Examiner requested an example to illustrate this advantage. One example is included in the specification.

Applicants' specification provides, as an example, a piece of source code to establish the declaration of an integer, I, and its initialization to zero. (Specification, page 10, lines 16-20). Considering this example, if two different programmers were to write pieces of source code implementing this idea, differences in each programmer's writing style might result in two different pieces of code. For example, the following pieces of source code might be written:

First Piece of Code	Second Piece of Code
INT I	INT I=0
I := 0	

Note that the source code written by the first programmer results in 2 lines of code, and the source code written by the second programmer results in 1 line of code. Thus, the substance of the code is the same but the statistics are different even in this simple example. If the two different pieces of source code are each parsed into tokens, the following tokens are obtained for each one:

First Piece of Code Tokens	Second Piece of Code Tokens
INT	INT
I	I
I	=
:=	0
0	

Accordingly, it can be seen that the first piece of code can be parsed into 5 tokens while the second piece of code can be parsed into 4 tokens. Thus, the generation of tokens reflects the stylistic differences between the two programmers. The lengthier and more complex the pieces of source code being created, the greater the difference between the number of tokens. As a result, a software tool that merely counts tokens, similar to those disclosed in *Scamp* and

Leonard, generates two different metric values for similarly written source code and, thus, is not able to account for these stylistic differences as well as the invention can.

The generation of a list of statements from each of the two pieces of source code, however, reduces the effect of stylistic differences on the metrics obtained for the pieces of source code. Returning to the example above, the following statements may be generated for each piece of code:

First Piece of Code Statements	Second Piece of Code Statements
INT I	INT I
I:=0	I = 0

Two statements are generated for each piece of code and the same number of statements is obtained in this example despite the stylistic differences of the programmers. Thus, an advantage provided by counting statements, rather than tokens, includes minimizing the effects of stylistic differences on the produced metrics. Comparisons "between a new version of a piece of software and a former version" may be produced to measure "how the new version has changed from the former version." (Specification, page 3, lines 17-20).

For at least these reasons, Applicants submit that neither *Scamp*, *Leonard*, nor their combination disclose, teach, or suggest a software counting tool operable to "create a list of statements in response to the token stream, at least some of the statements comprising a combination of two or more tokens," and "generate a count value in response to the list of statements," as recited in Applicants' Claim 1. Accordingly, Applicants respectfully request reconsideration and allowance of Claim 1, together with Claims 2-12 that depend from Claim 1.

Independent Claims 13, 23, 29, 35, and 37 recite certain limitations that are similar to those discussed above with regard to Claim 1. Accordingly, for reasons similar to those discussed above with regard to Claim 1, Applicants respectfully submit that the *Scamp-Leonard-Duan* combination does not disclose, teach, or suggest each and every element recited in Applicants' Claims 13, 23, 29, 35, and 37. Claims 14-22, 24-28, 30-34, and 36 depend directly or indirectly upon Claims 13, 23, 29, and 35, respectively. Thus, for the same reasons that independent Claims 13, 23, 29, and 35 are allowable, these dependent claims are also allowable.

One of Ordinary Skill in the Art Would not have been Motivated to Make the Proposed Combinations of References

Assuming for purposes or argument that the proposed combination discloses the limitations of Applicants' claims, which Applicants dispute, it would not have been obvious to one skilled in the art to make the proposed combinations of references. The Examiner has not provided adequate evidence of the required motivation or suggestion to make the proposed combination. For example, with regard to the combination of *Scamp* with *Leonard*, the Examiner merely speculates "it would have been obvious" to modify *Scamp* "so that line of code technique as suggested by Leonard can be added because line of codes can be a factor determining how to predict resources, maintain, manage, or improve the state of a software product according to *Leonard's* approach of having a database for a life-cycle process associated with configuration management activity." (Office Action, page 4). The Examiner has not shown any motivation to combine and instead simply relies upon hindsight.

As discussed above, *Scamp* discloses a software compiler for analyzing the tokens in an existing file of source code. The objective of *Scamp* is to identify the frequency and location of various tokens with the existing source code. The objective of *Leonard*, on the other hand, is the generation of a predictor metric that may have a "strong correlation to some later result." (Column 3, lines 66-67). *Leonard* provides metrics for predicted source code <u>before</u> it is written rather than for written and existing source code. Since *Leonard* is concerned with source code that is not yet written, the stated objective of *Leonard*, which the Examiner relies on to make the proposed combination, is not relevant to the system of *Scamp*.

For at least these reasons, Applicants respectfully submit that the proposed combinations of references are improper. Accordingly, the rejection of Applicants' claims over the proposed combinations of references should be withdrawn.

CONCLUSION

Applicants have made an earnest attempt to place this case in condition for immediate allowance. For the foregoing reasons and for other reasons clear and apparent, Applicants respectfully request reconsideration and allowance of the pending claims.

Applicants do not believe any fees are due. However, the Commissioner is hereby authorized to charge any additional fees or credit any overpayment to Deposit Account No. 05-0765 of Electronic Data Systems Corporation.

If there are matters that can be discussed by telephone to advance prosecution of this application, Applicants invite the Examiner to contact its attorney at (214) 953-6809.

Respectfully submitted,

Baker Botts L.L.P.
Attorneys for Applicants

Jenni R. Moen

Jenni R. Moen
Reg. No. 52,038

Dated: September 30, 2004

CORRESPONDENCE ADDRESS:

at Customer No. 35005